# CSE331: Microprocessor Interfacing and Embedded Systems

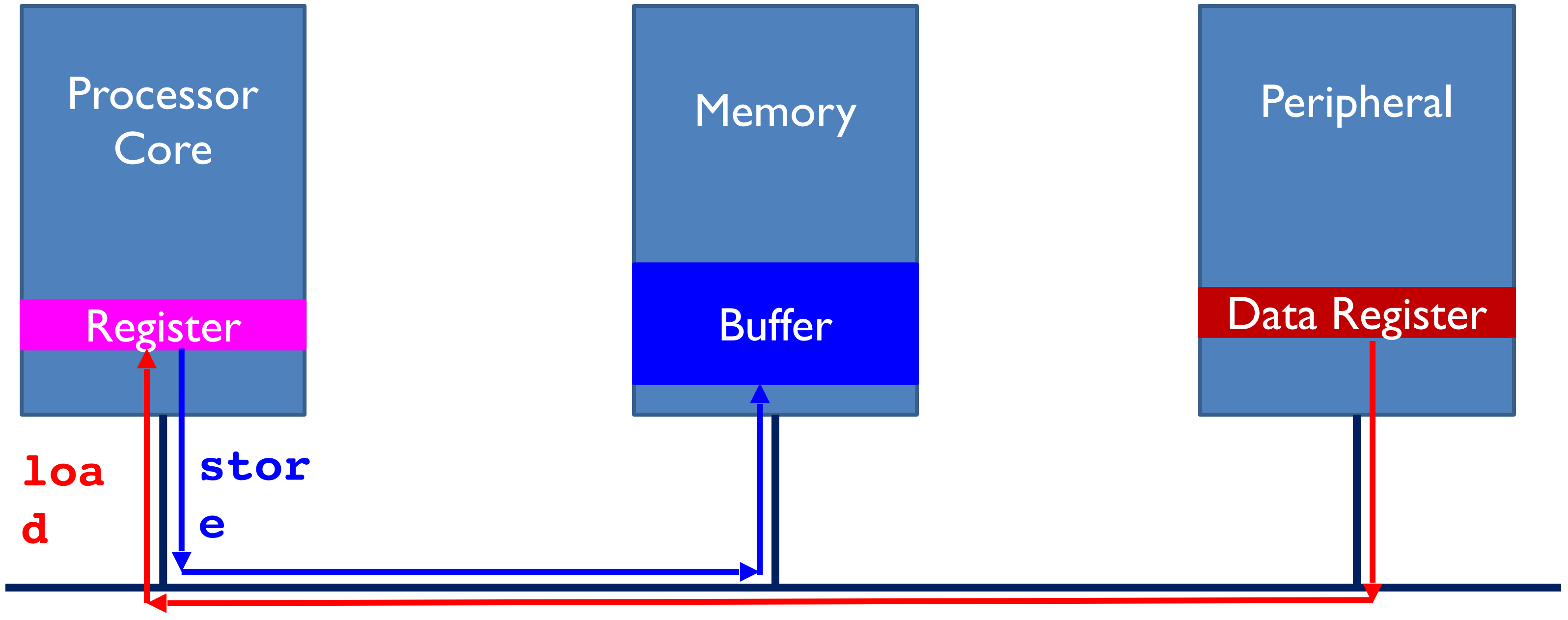
## Lecture#21: Direct Memory Access (DMA)
## Chapter 19

Summer, 2023

Mohammad A. Qayum, Ph.D.
ECE, NSU, Dhaka, Bangladesh

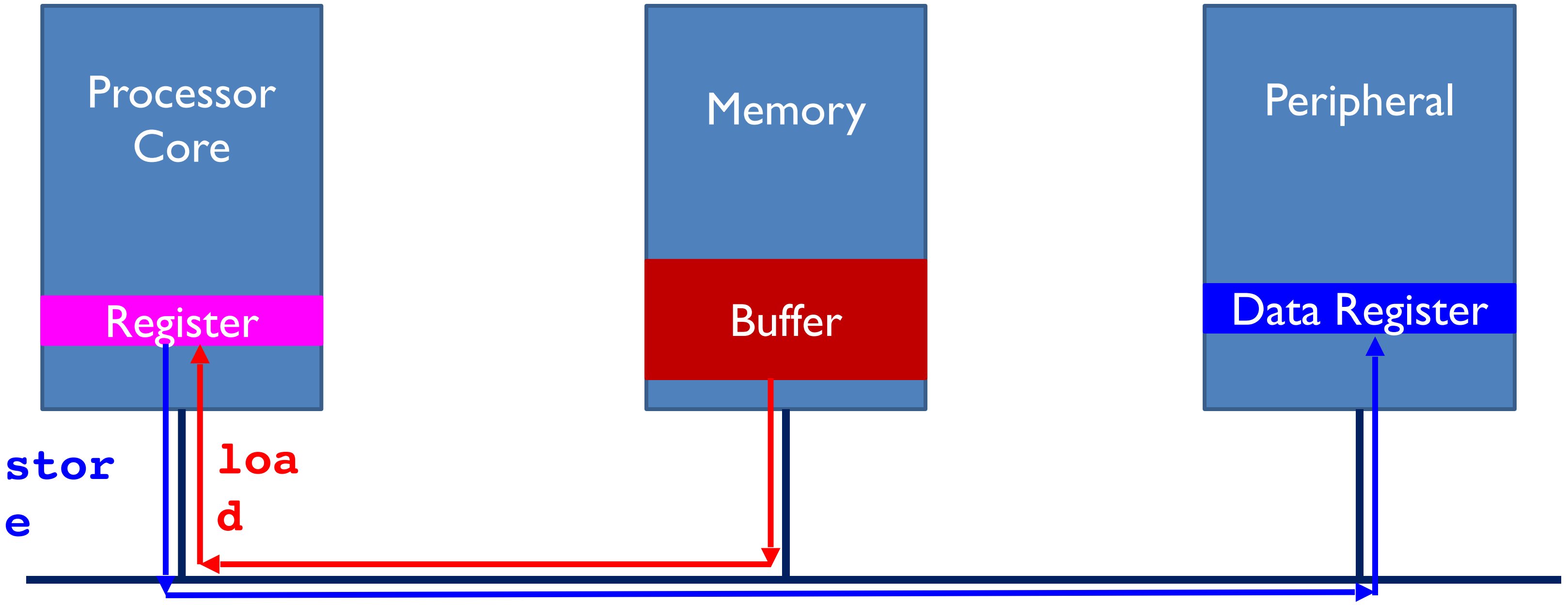
## Moving Data between Peripheral and Memory

- Copy data from peripheral to buffer



## Moving Data between Peripheral and Memory

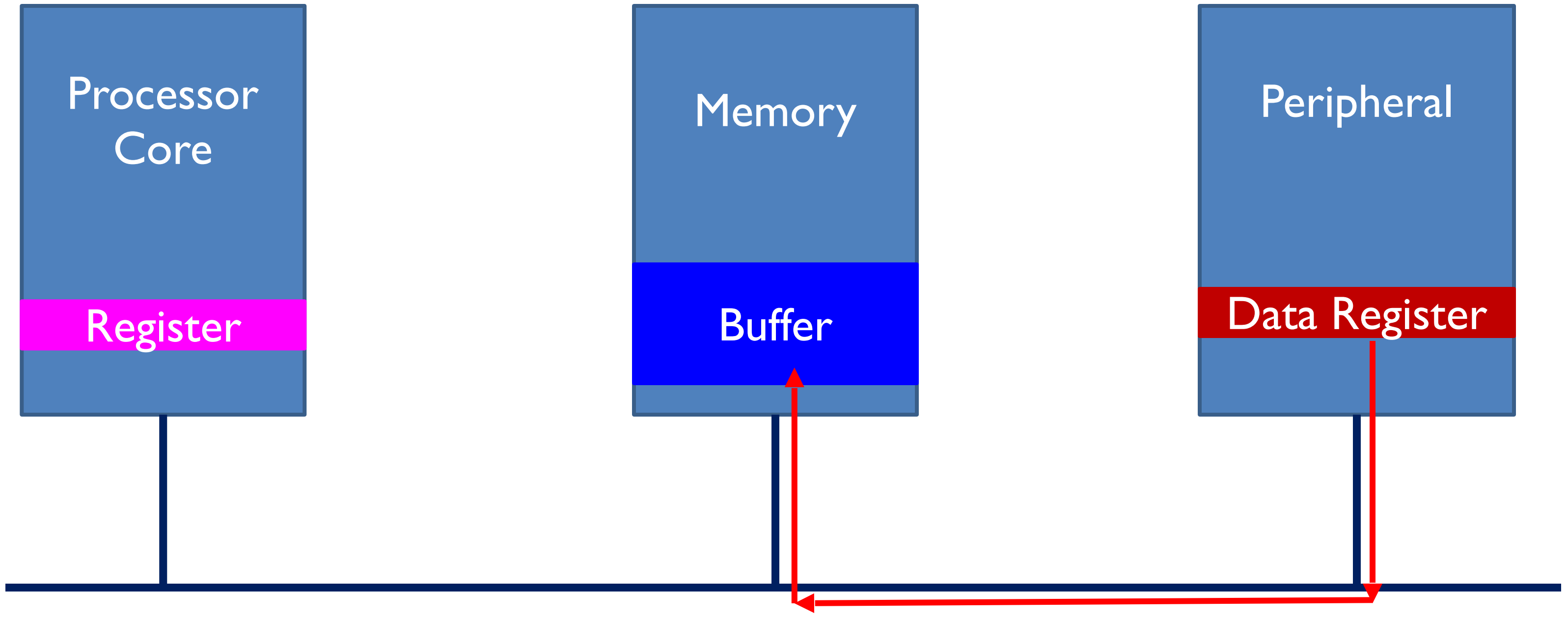
- Copy data from buffer to peripheral



## DMA

- Copy data from peripheral to buffer

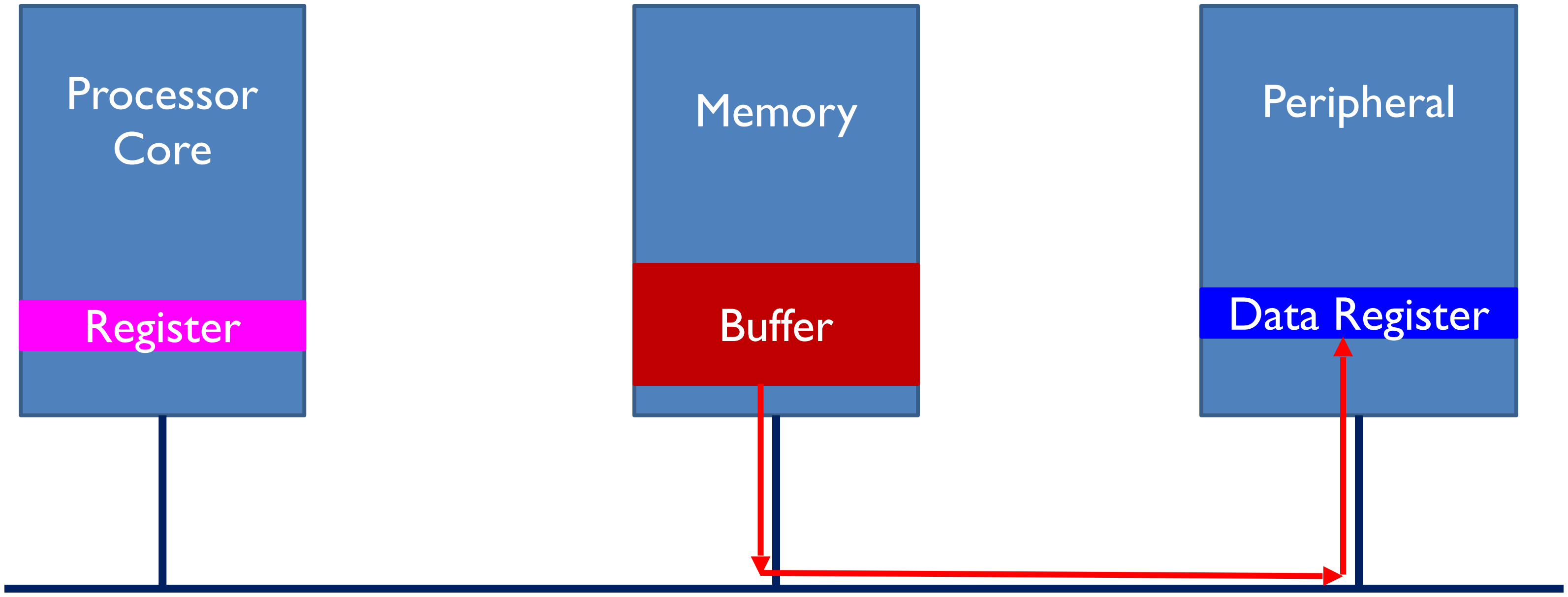
***DMA releases CPU from moving data***



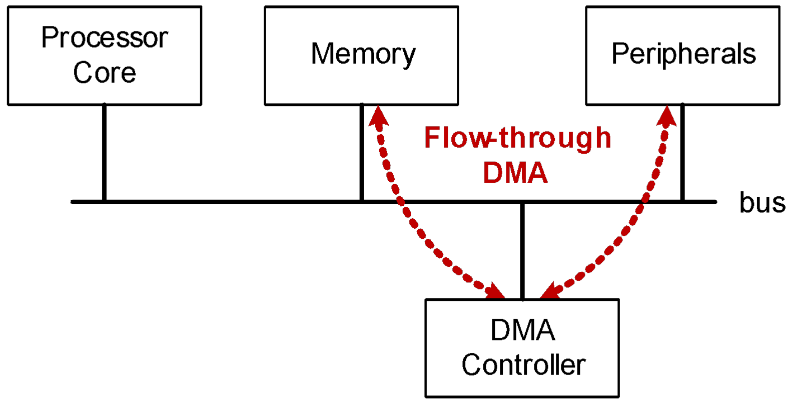
# DMA

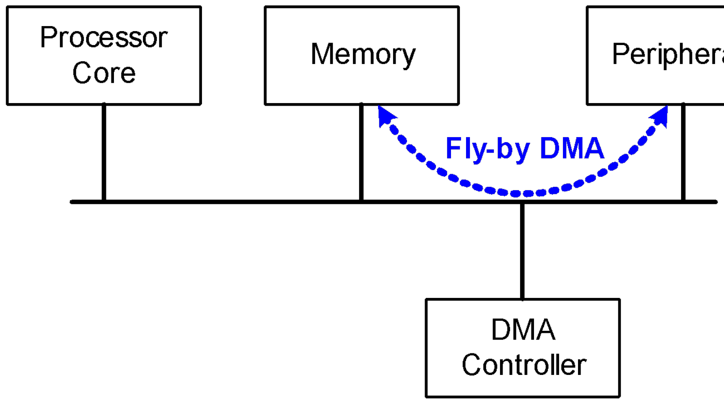
- Copy data from buffer to peripheral

***DMA releases CPU from moving data***



## DMA: Flow-though *vs* Fly-by





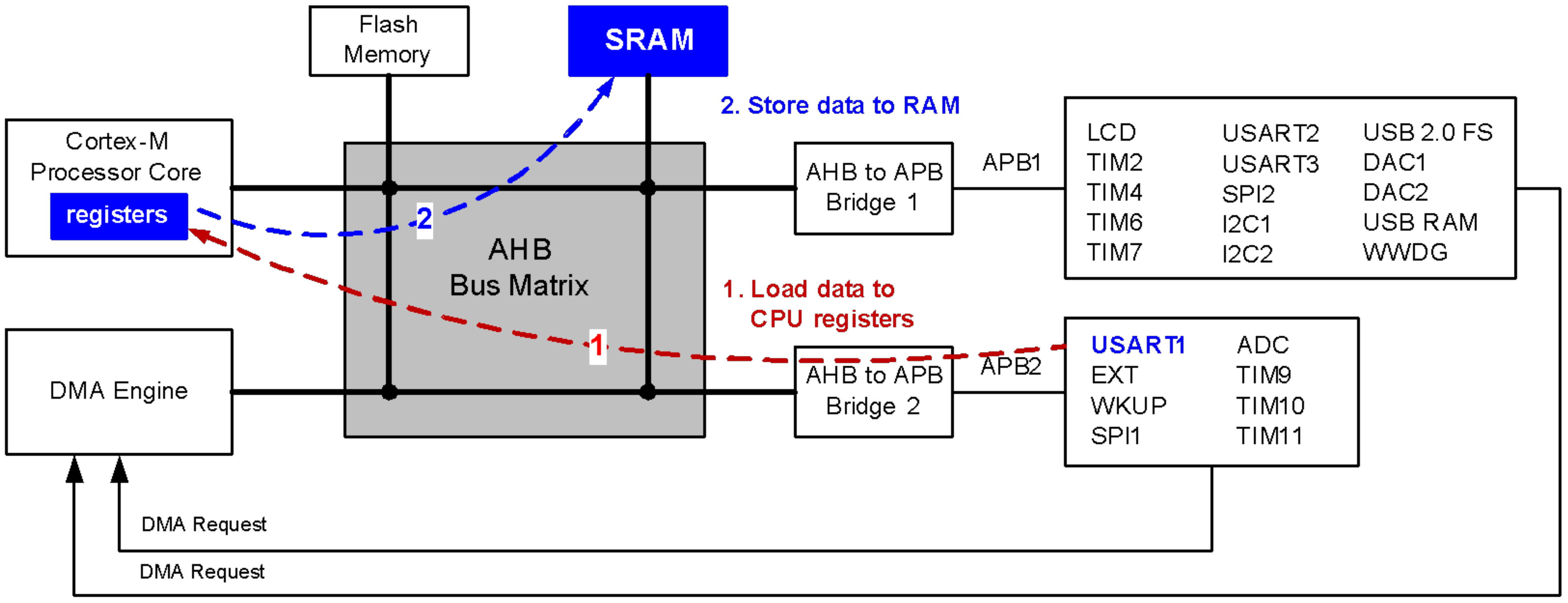
# DMA Channels

- DMA releases CPU from moving data
  - between peripherals and memory, or
  - between one peripheral and another peripheral.
- DMA uses bus matrix to allow concurrent transfers

| Peripherals | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7 |
|---|---|---|---|---|---|---|---|
| **ADC1** | ADC1 | | | | | | |
| **SPI** | | SPI1_RX | SPI1_TX | SPI2_RX | SPI2_TX | | |
| **USART** | | USART3_TX | USART3_RX | USART1_TX | USART1_RX | USART2_RX | USART2_TX |
| **I2C** | | | | I2C2_TX | I2C2_RX | I2C1_TX | I2C1_RX |
| **TIM2** | TIM2_CH3 | TIM2_UP | | | TIM2_CH1 | | TIM2_CH2<br>TIM2_CH4 |
| **TIM3** | | TIM3_CH3 | TIM3_CH4<br>TIM3_UP | | | TIM3_CH1<br>TIM3_TRIG | |
| **TIM4** | TIM4_CH1 | | | TIM4_CH2 | TIM4_CH3 | | TIM4_UP |
| **TIM6<br>DAC_Ch1** | | TIM6_UP<br>DAC_Ch1 | | | | | |
| **TIM7<br>DAC_CH2** | | | TIM7_UP<br>DAC_CH2 | | | | |

# Programmed I/Os

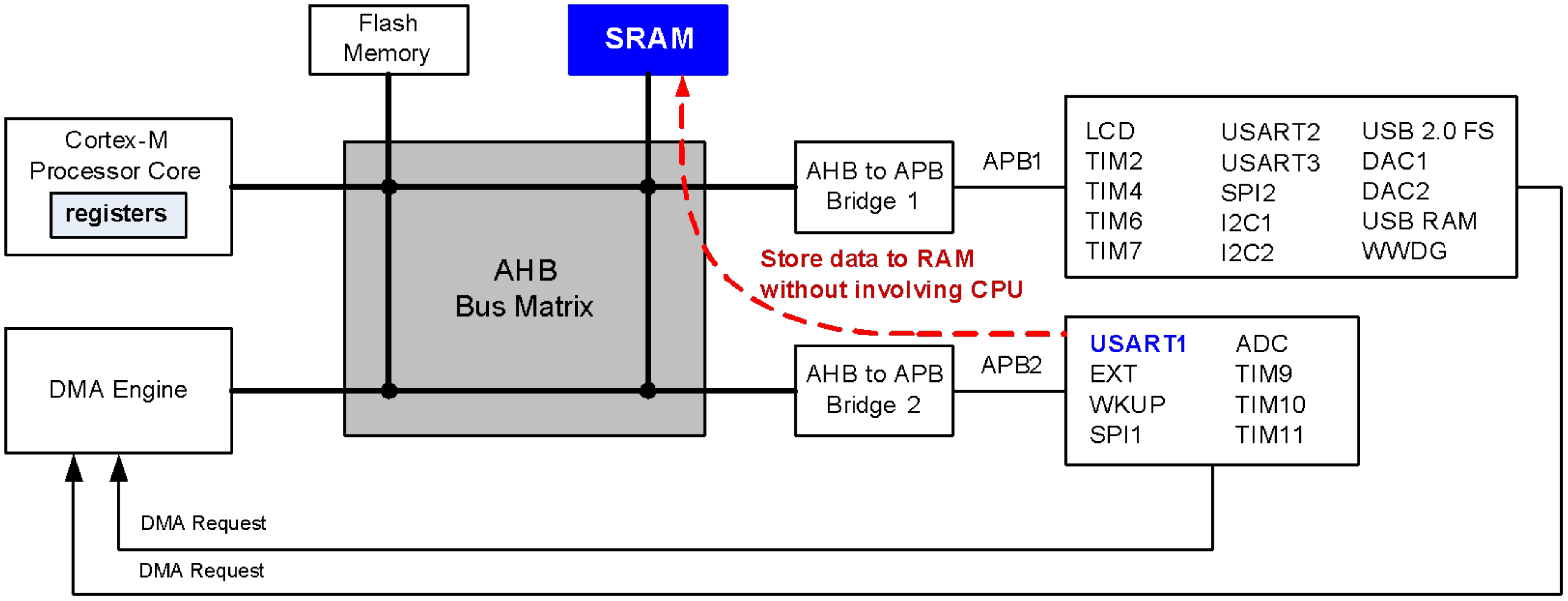
- Processor executes a lot Loads/Stores to move data
- High overhead and slow



Receiving data from USART serial port without using DMA

# DMA Sets Core Free

- CPU delegates reads/writes to DMA controller
- Low overhead and fast



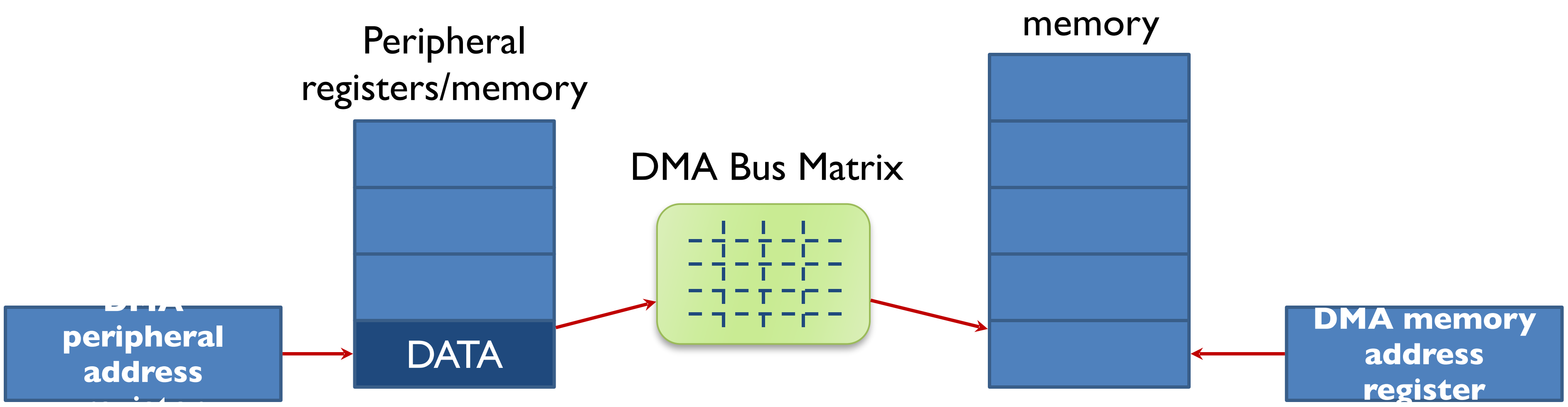
Receiving data from USART serial port using DMA

## DMA Controller

- Basic Procedures
  - DMA device requests bus
  - CPU grants bus request
  - CPU takes its signals to HiZ
- Key DMA Controller Registers
  - DMA memory address register (CMAR) (C stands for Channel)
  - DMA peripheral address register (CPAR)
  - DMA number of data register (CNDTR)
  - DMA configuration register (CCR)
- DMA are often used together with interrupts

# DMA Mode: Incremental Mode

DIR: Data transfer direction: 0 = Read from peripheral; 1 = Read from memory



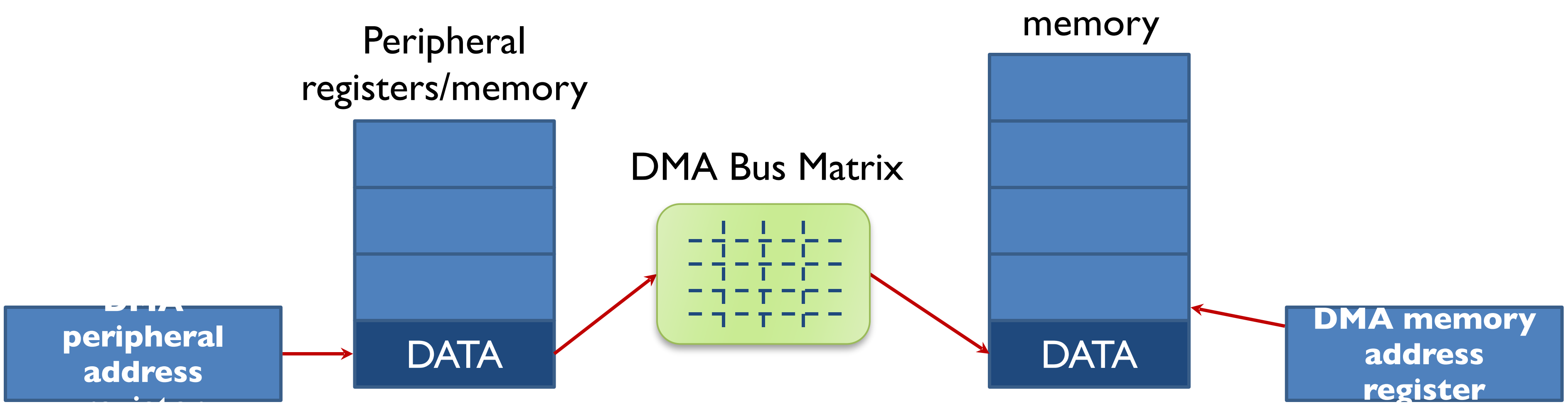
non-incremental mode

**DIR = 0**
**CNDTR = 2**

incremental mode

# DMA Mode: Incremental Mode

DIR: Data transfer direction: 0 = Read from peripheral; 1 = Read from memory


Peripheral registers/memory
memory
DMA Bus Matrix
DMA peripheral address register
DATA
DATA
DMA memory address register

non-incremental mode

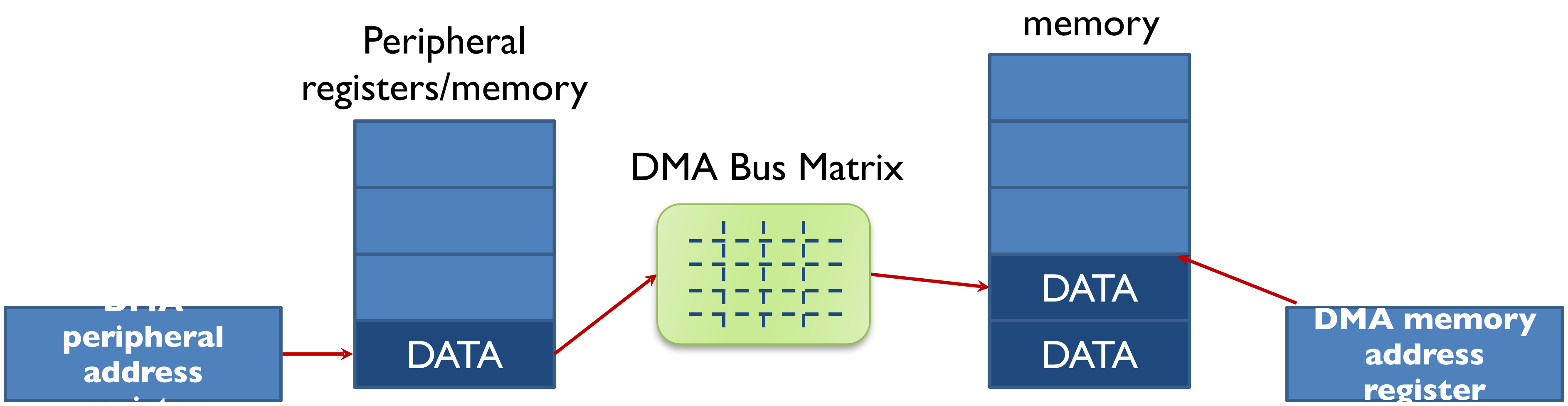
**DIR = 0**
**CNDTR = 1**

incremental mode

**1st DMA Transfer**

# DMA Mode: Incremental Mode

DIR: Data transfer direction:   0 = Read from peripheral;  1 = Read from memory


Peripheral registers/memory
DMA Bus Matrix
memory
DMA peripheral address register
DATA
DATA
DATA
DMA memory address register

non-incremental mode

**DIR = 0**
**CNDTR = 0**

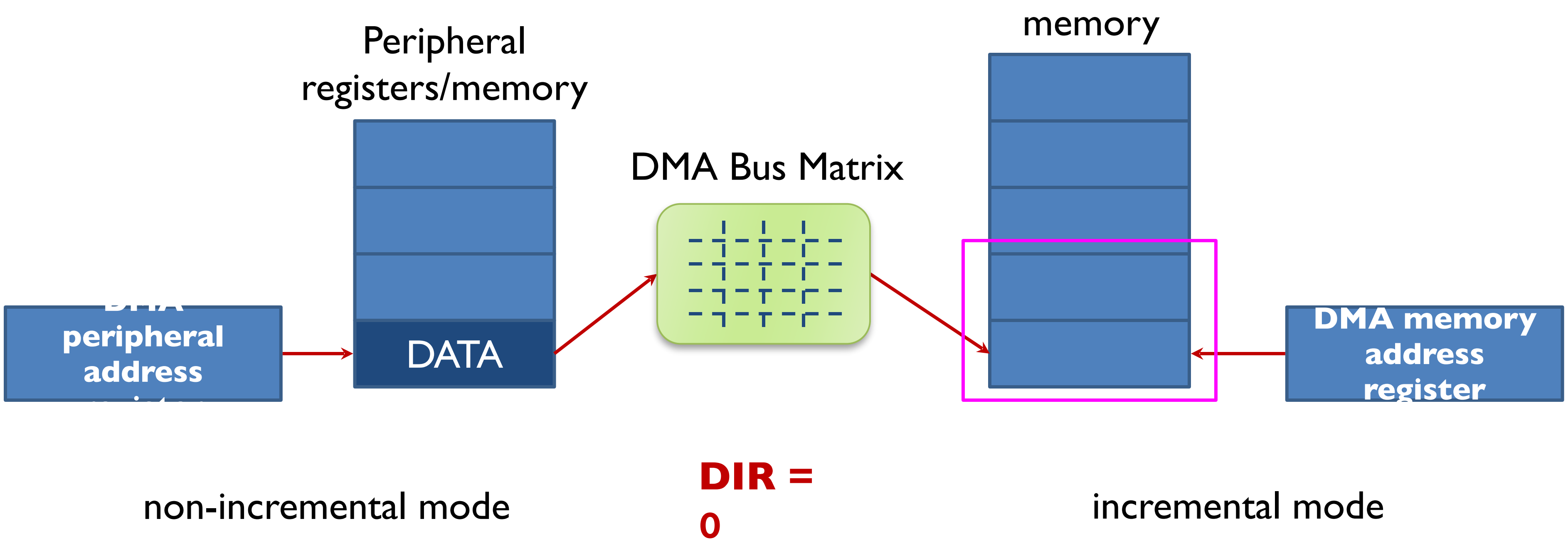
incremental mode

**2nd DMA Transfer**

**DMA stops since CNDTR is zero now.**

# DMA Mode: Circular Mode

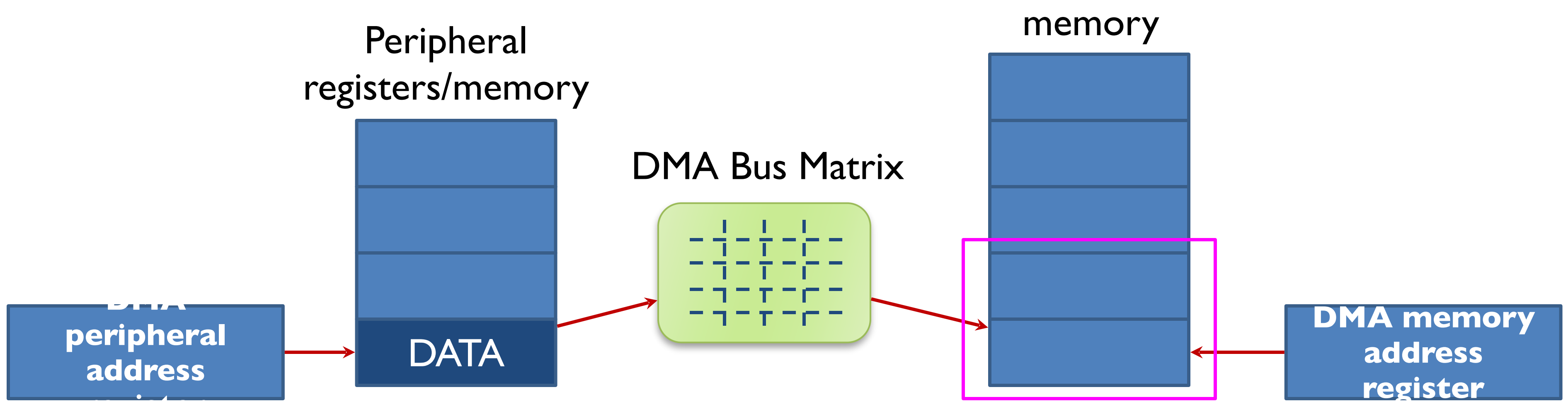
DIR: Data transfer direction: 0 = Read from peripheral; 1 = Read from memory



- **Circular Mode**
  - handle circular buffers and continuous data flows
  - The number of data to be transferred (CNDTR) is automatically reloaded and DMA requests continue to be served

# DMA Mode: Incremental Mode

DIR: Data transfer direction: 0 = Read from peripheral; 1 = Read from memory



Peripheral registers/memory
DMA Bus Matrix
memory
DMA peripheral address register
DATA
DMA memory address register

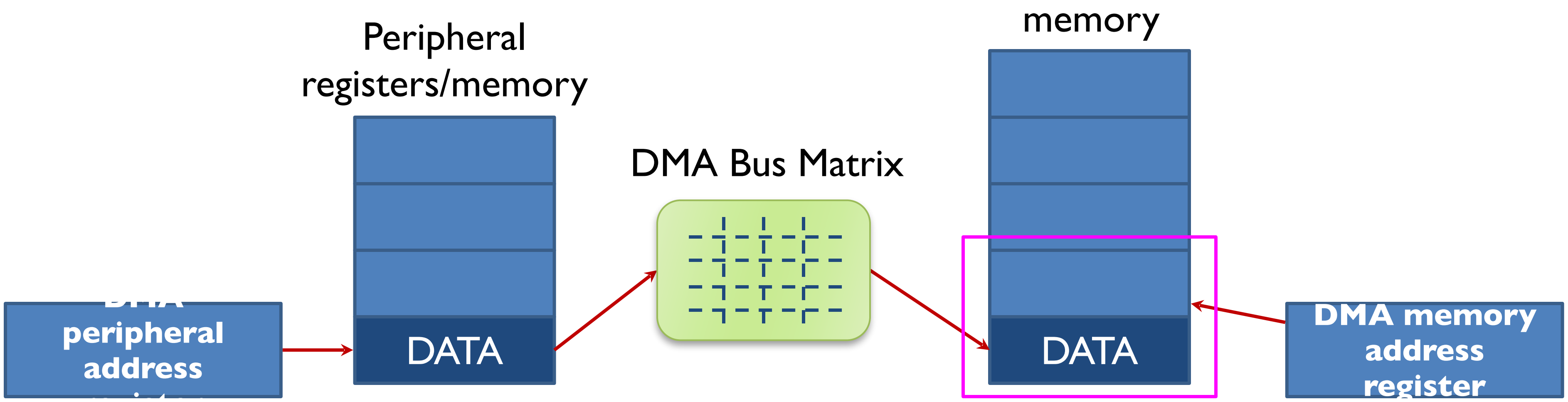
non-incremental mode

**DIR = 0**
**CNDTR = 2**
**CIR = 1**

incremental mode

# DMA Mode: Incremental Mode

DIR: Data transfer direction:   0 = Read from peripheral;  1 = Read from memory


Peripheral registers/memory
DMA Bus Matrix
memory
DMA peripheral address register
DATA
DATA
DMA memory address register

non-incremental mode

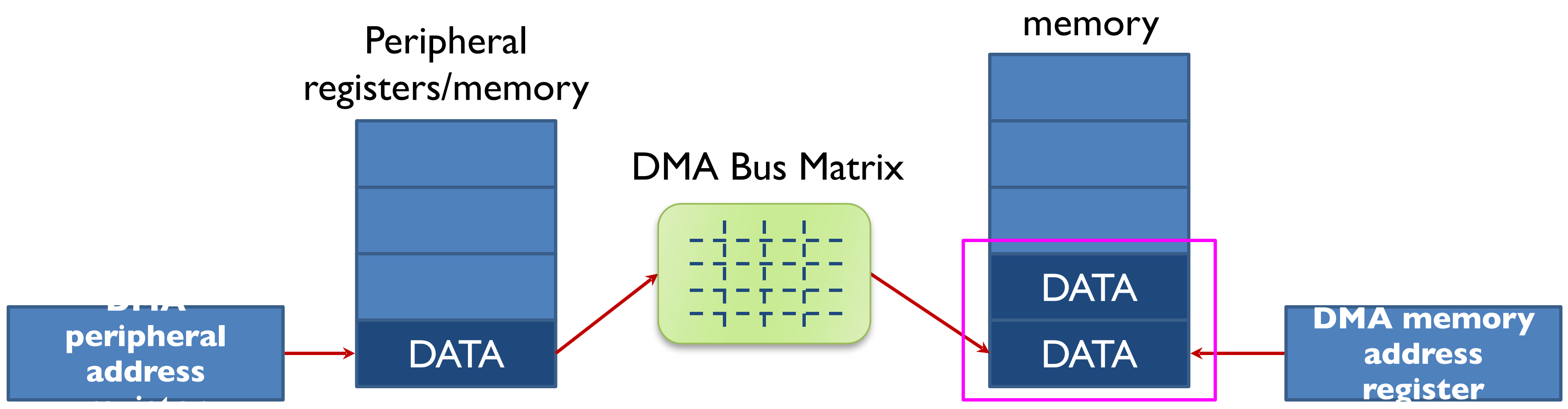
**DIR = 0**
**CNDTR = 1**
**CIR = 1**

**1st DMA Transfer**

incremental mode

# DMA Mode: Incremental Mode

DIR: Data transfer direction: 0 = Read from peripheral; 1 = Read from memory



Peripheral registers/memory
DMA peripheral address register
DATA
DMA Bus Matrix
memory
DATA
DATA
DMA memory address register

non-incremental mode

**DIR = 0**
**CNDTR = 0**
**CIR = 1**

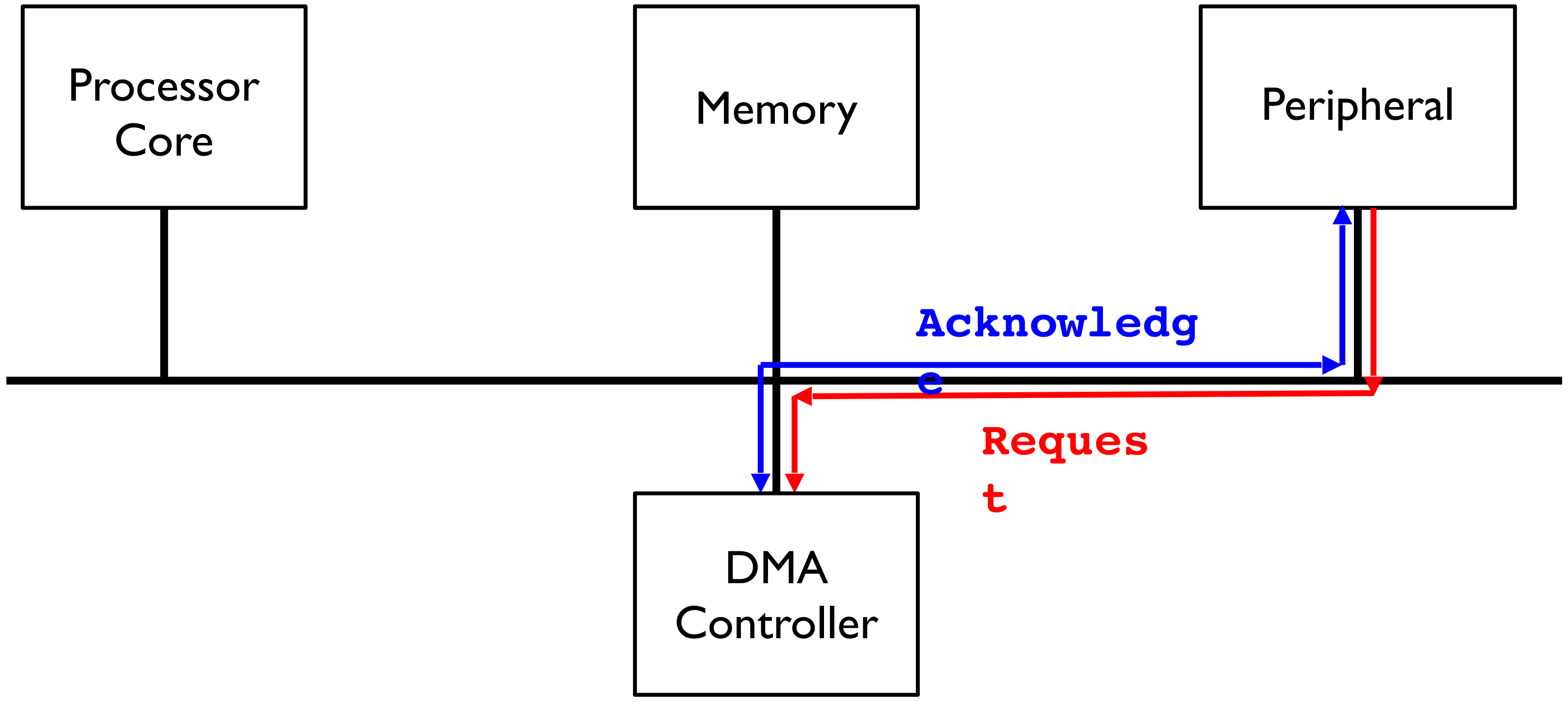
**2nd DMA Transfer**

incremental mode

**DMA resets and continues to run!**

## DMA Request

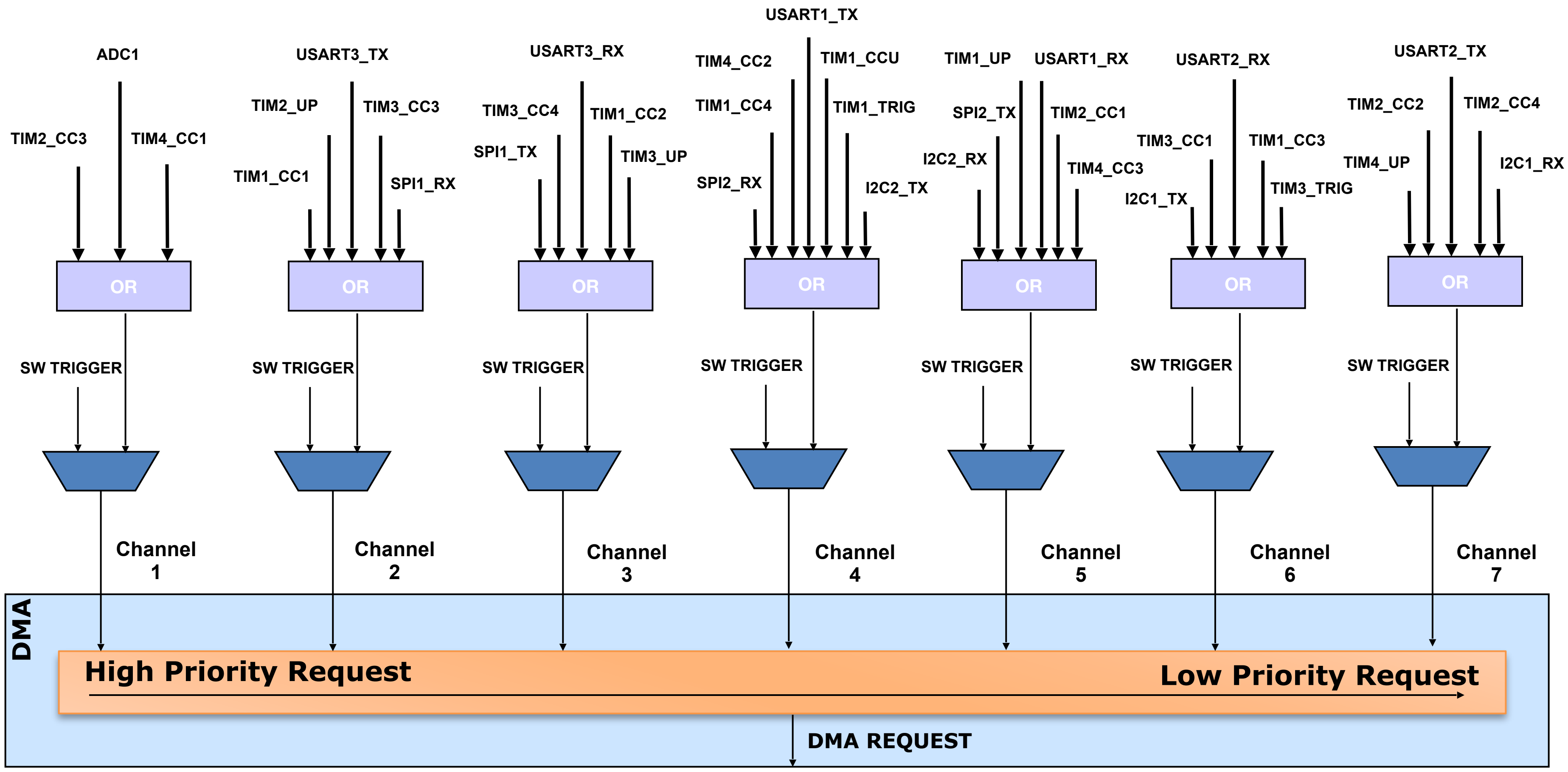
- When does the next DMA transfer start?
  - When the peripheral is ready to send or receive data, the peripheral will generate a DMA request signal to the DMA controller to request a data transfer.



## DMA Interrupts

- Programmable and Independent source and destination transfer data size: Byte, Halfword or Word
- Three event flags: DMA Half Transfer, DMA Transfer complete and DMA Transfer Error
- Software programmable priorities: Very high, High, Medium or Low

# DMA Request Mapping



*From stm*

## DMA Summary

- Without DMA, CPU has to execute many load and store instructions, leading to slower performance.
- DMA, which makes an automatic data transfer when received a DMA request without involving CPU, accelerates the overall performance.

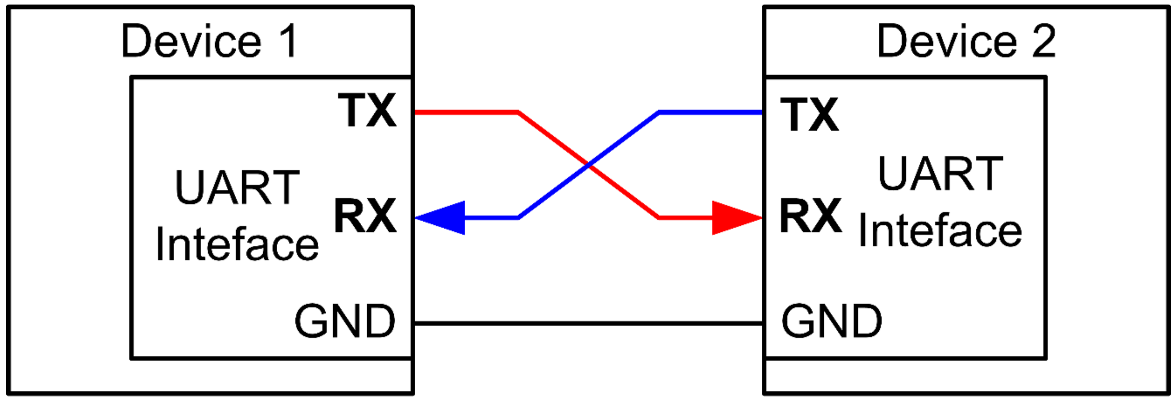
# CSE331: Microprocessor Interfacing and Embedded Systems

## Lecture#24: Serial Communication: UART (Chapter 22)

Mohammad A. Qayum, Ph.D.
ECE, NSU, Dhaka, Bangladesh

## Universal Asynchronous Receiver and Transmitter (UART)

- Universal
  - UART is programmable.
- Asynchronous
  - Sender provides no clock signal to receivers



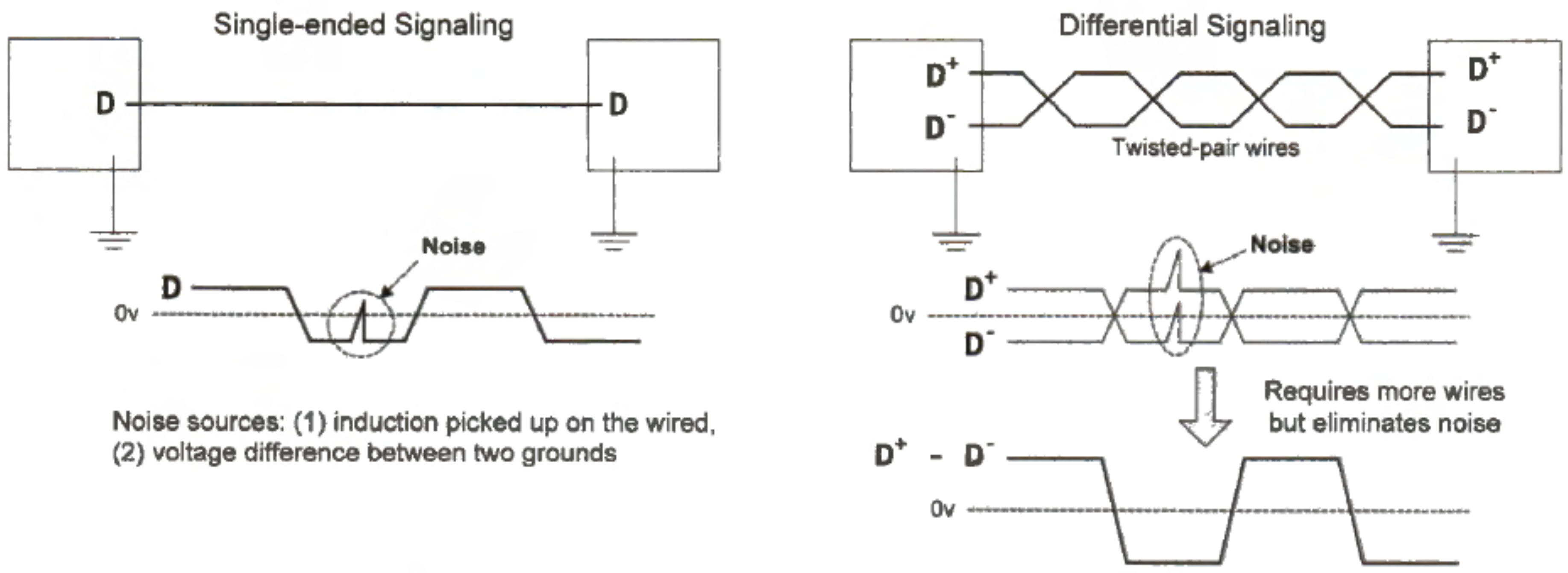


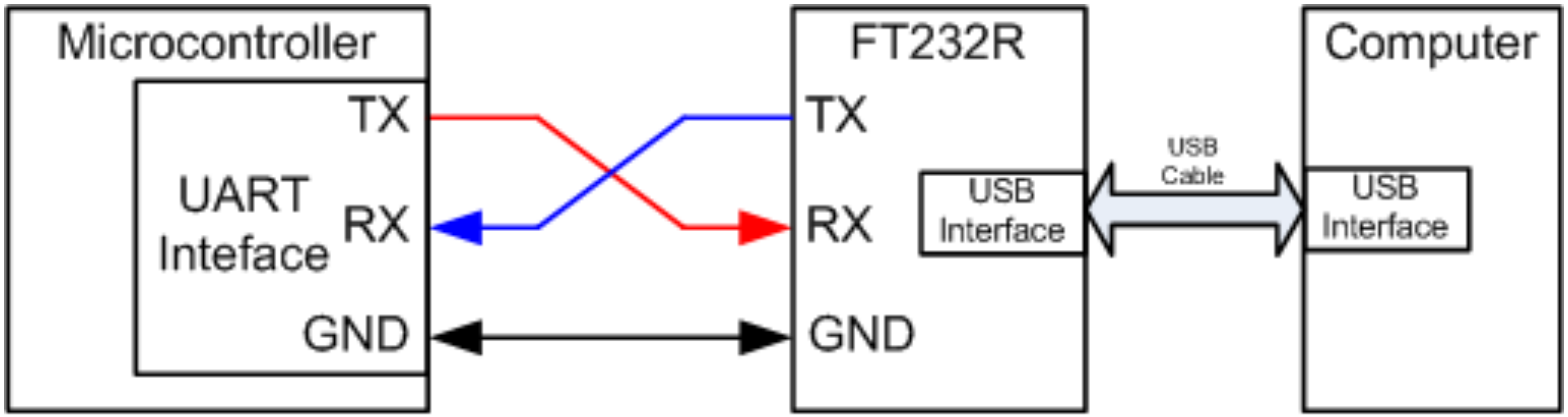
Figure 22-4. Comparison of single-ended and differential signaling

## Voltage Levels

| Standard | Voltage signal | Max distance | Max speed | Number of devices supported per port |
|---|---|---|---|---|
| RS-232 | Single end ( logic 1: +5 to +15V, logic 0: -5 to -15 V) | 100 feet | 115Kbit/s | 1 master, 1 receiver |
| RS-422 | Differential (-6V to +6V) | 4000 feet | 10Mbit/s | 1 master, 10 receivers |
| RS-485 | Differential (-7V to +12V) | 4000 feet | 10Mbit/s | 32 masters, 32 receivers |

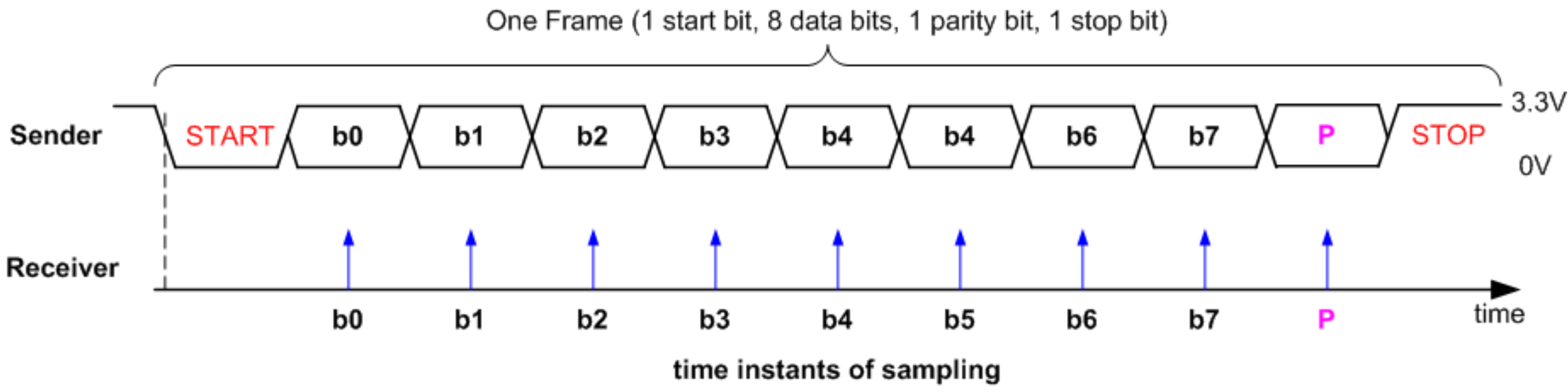
## Connecting to PC

- FT232R converts the UART port to a standard USB interface



USB to serial UART

## Data Frame


One Frame (1 start bit, 8 data bits, 1 parity bit, 1 stop bit)
Sender
START
b0
b1
b2
b3
b4
b4
b6
b7
P
STOP
3.3V
0V
Receiver
b0
b1
b2
b3
b4
b5
b6
b7
P
time
time instants of sampling

**Tolerate 10% clock shift during transmission**

- Sender and receiver uses the same transmission speed
- Data frame
  - One start bit
  - Data (LSB first or MSB, and size of 7, 8, 9 bits)
  - Optional parity bit
  - One or two stop bit

## Baud Rate

- Historically used in telecommunication to represent the number of pulses physically transferred per second
- In digital communication, baud rate is the number of bits physically transferred per second
- Example:
  - Baud rate is 9600
  - each frame: a start bit, 8 data bits, a stop bit, and no parity bit.
  - Transmission rate of actual data

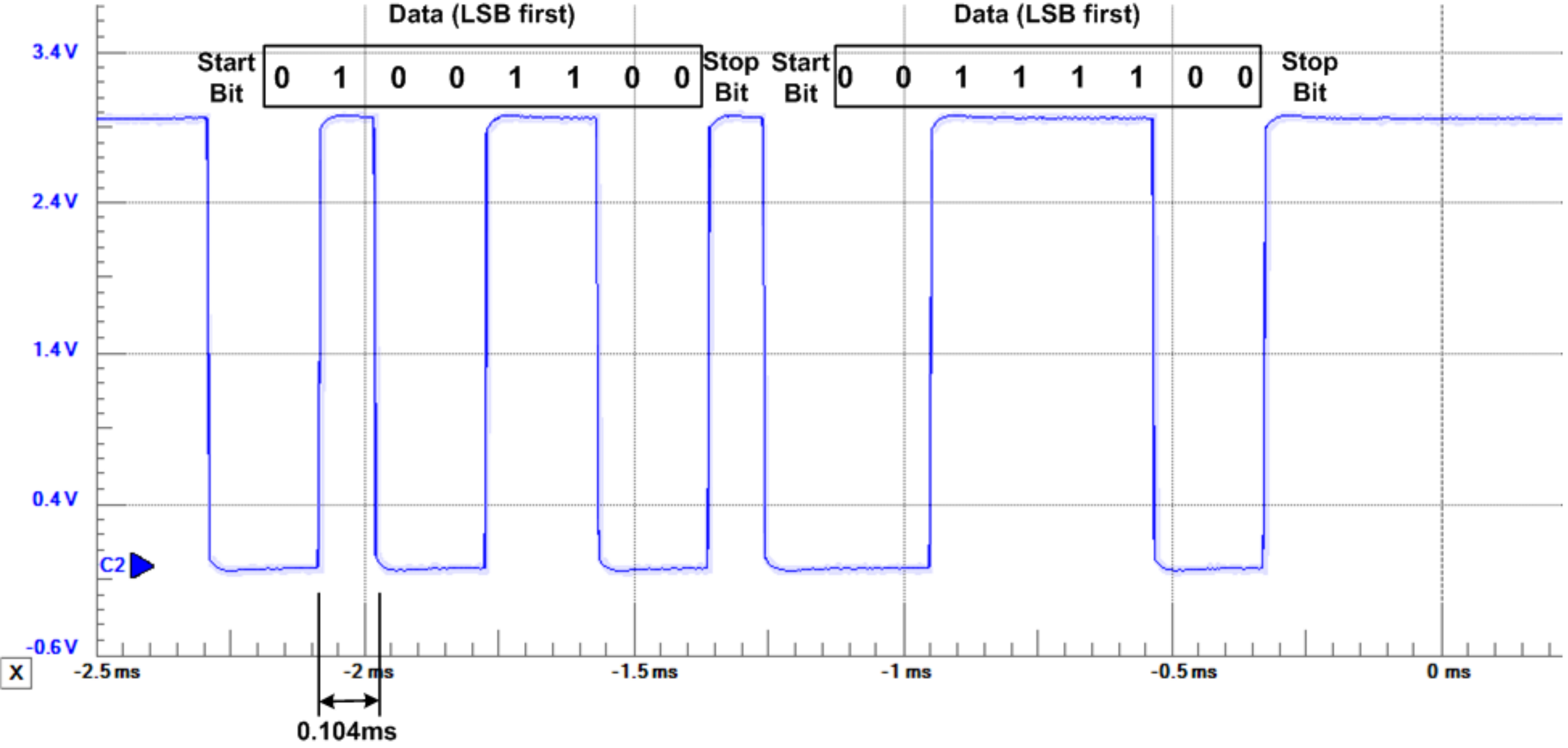
    ~~9600/8 = 1200 bytes/second~~

    9600/(1 + 8 + 1) = 960 bytes/second
  - The start and stop bits are the protocol overhead

## Error Detection

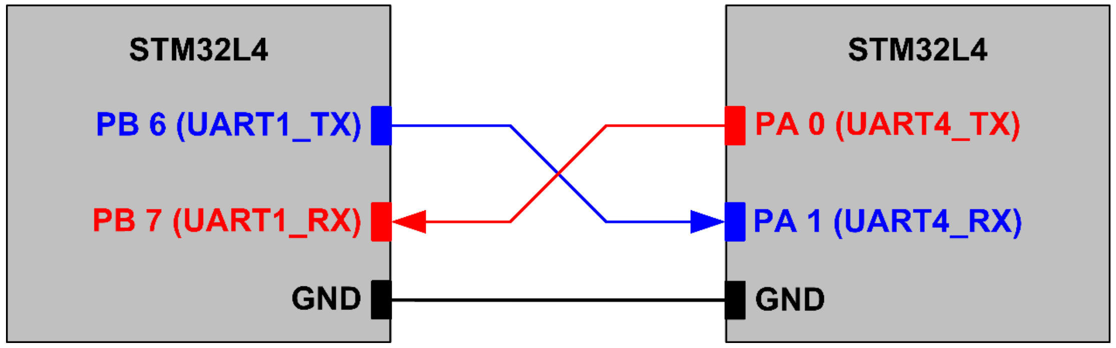
- Even Parity: total number of "1" bits in data and parity should be even
- Odd Parity: total number of "1" bits in data and parity should be odd
- Example:  Data = 10101011 (five "1" bits)
  - The parity bit should be 0 for odd parity and 1 for even parity
- This can detect single-bit data corruption

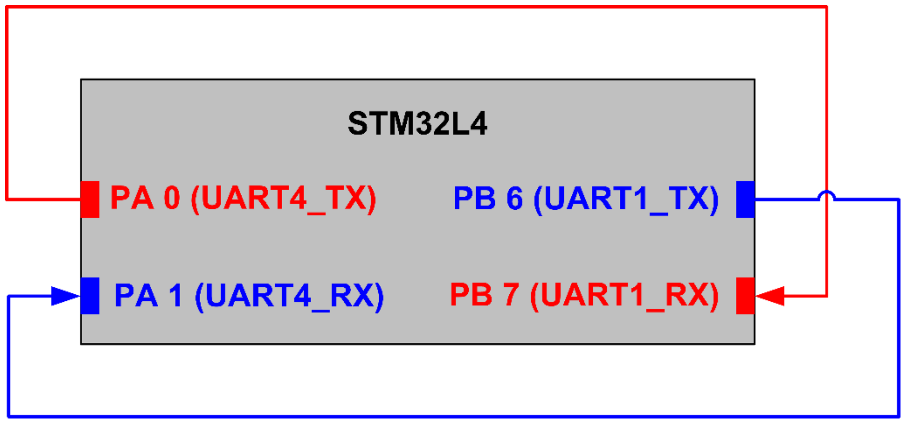
# Transmitting 0x32 and 0x3C



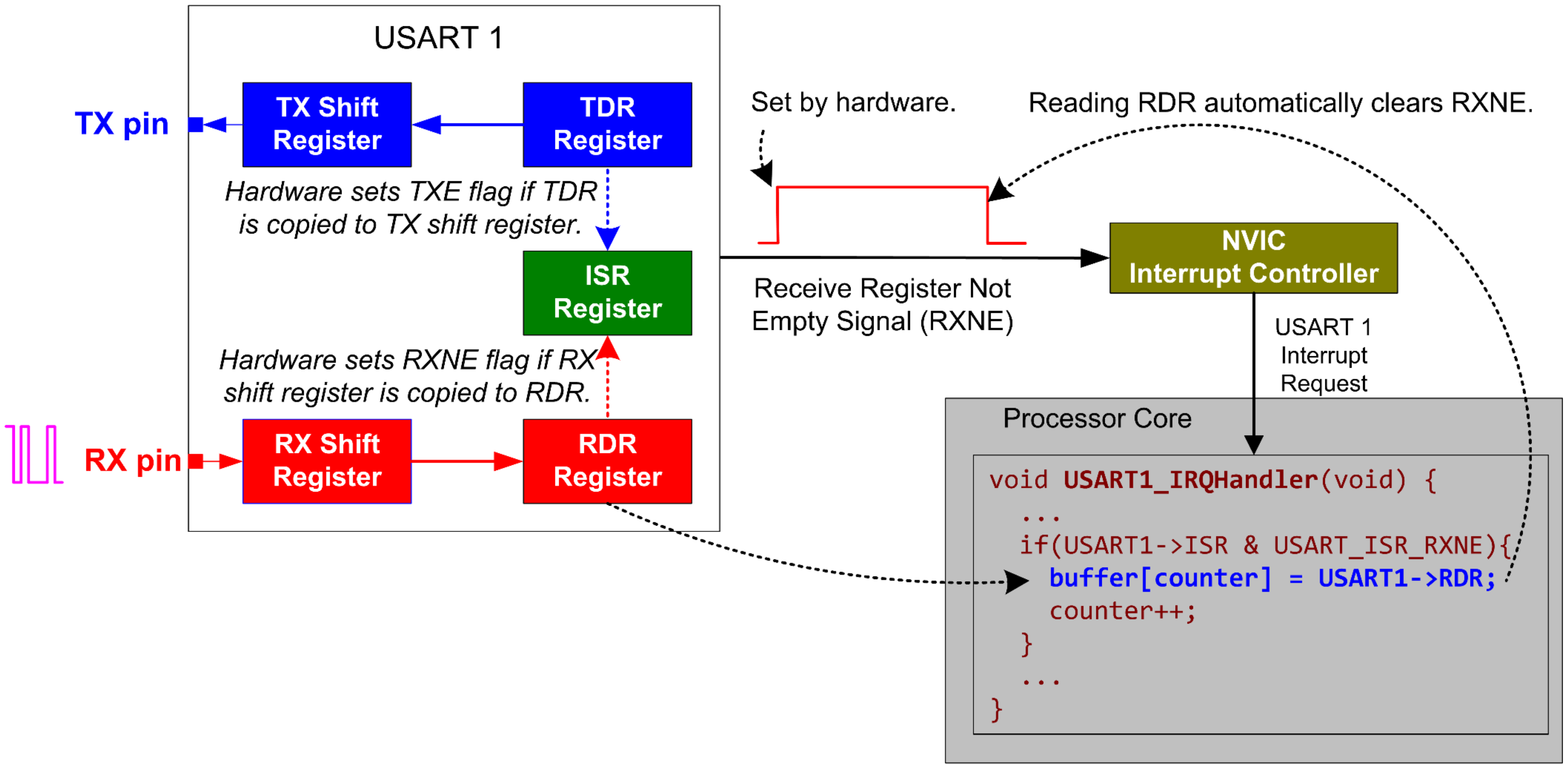
**1 start bit, 1 stop bit, 8 data bits, no parity, baud rate = 9600**

# UART Connection





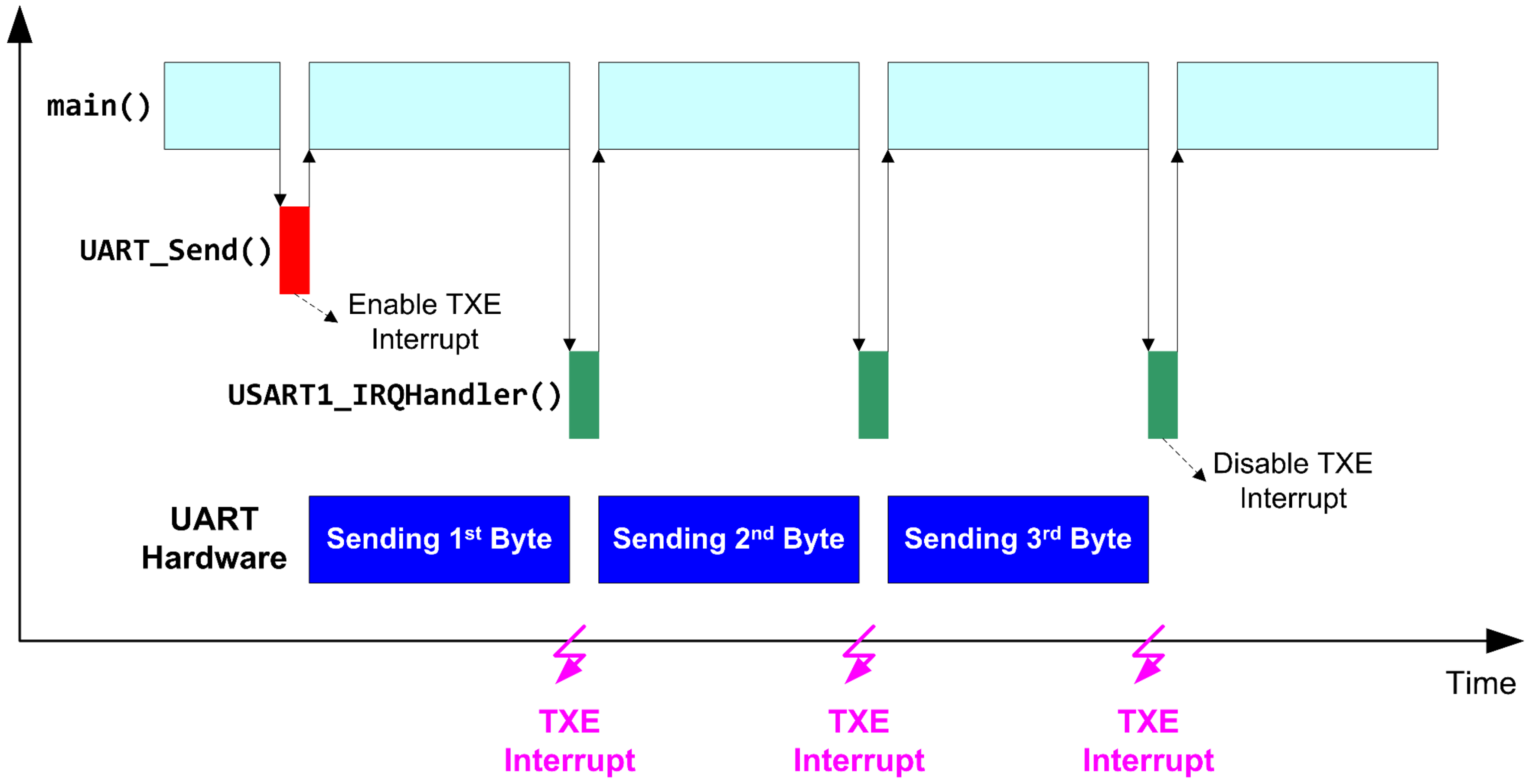
# UART Interrupt: Receiving Data



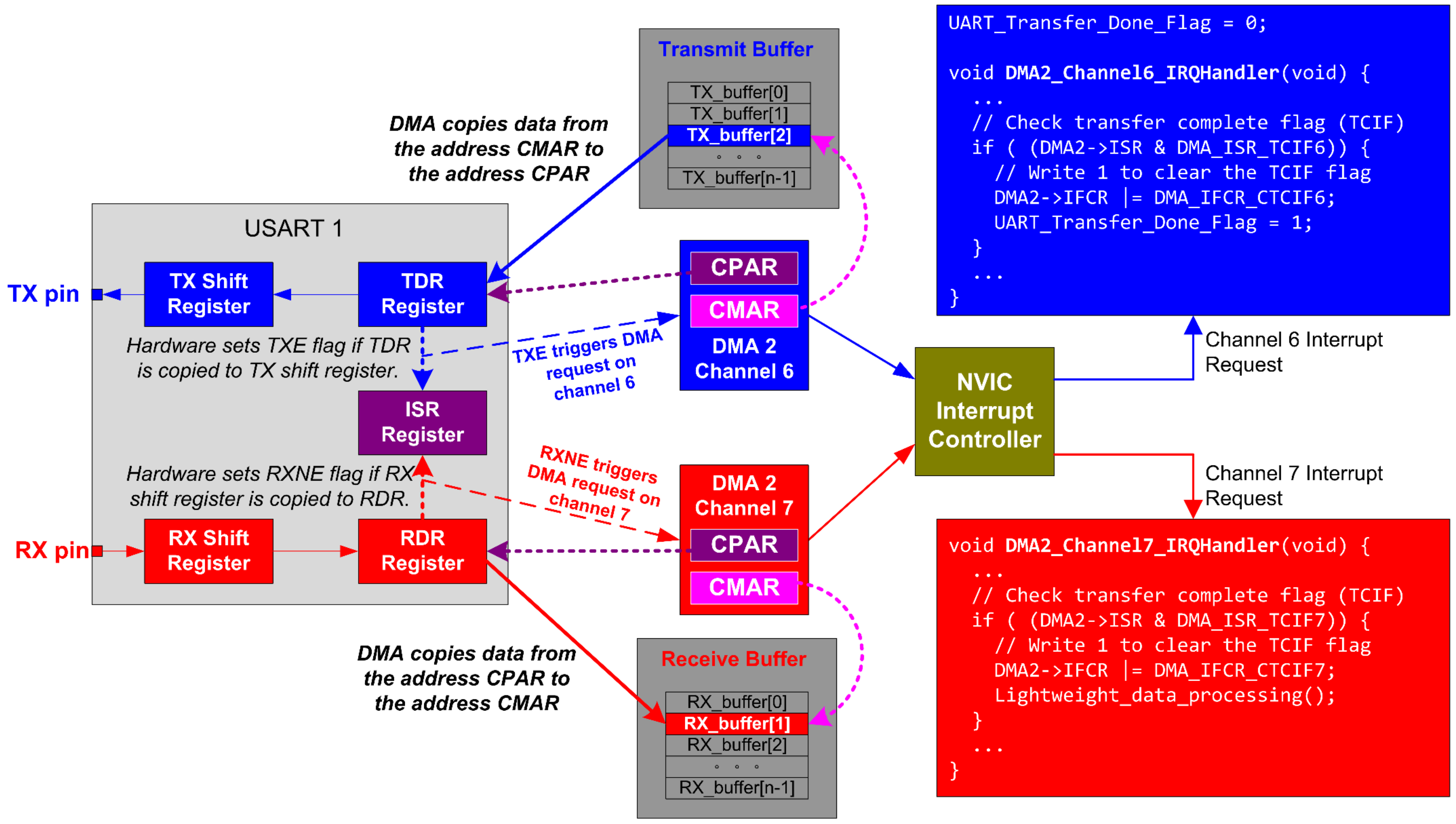
USART 1
TX pin
TX Shift Register
TDR Register
Hardware sets TXE flag if TDR is copied to TX shift register.
ISR Register
Hardware sets RXNE flag if RX shift register is copied to RDR.
RX pin
RX Shift Register
RDR Register
Set by hardware.
Reading RDR automatically clears RXNE.
NVIC Interrupt Controller
Receive Register Not Empty Signal (RXNE)
USART 1 Interrupt Request
Processor Core

```
void USART1_IRQHandler(void) {
  ...
  if(USART1->ISR & USART_ISR_RXNE){
    buffer[counter] = USART1->RDR;
    counter++;
  }
  ...
}
```

# UART Interrupt: Receiving Data



# UART DMA: Receiving & Sending



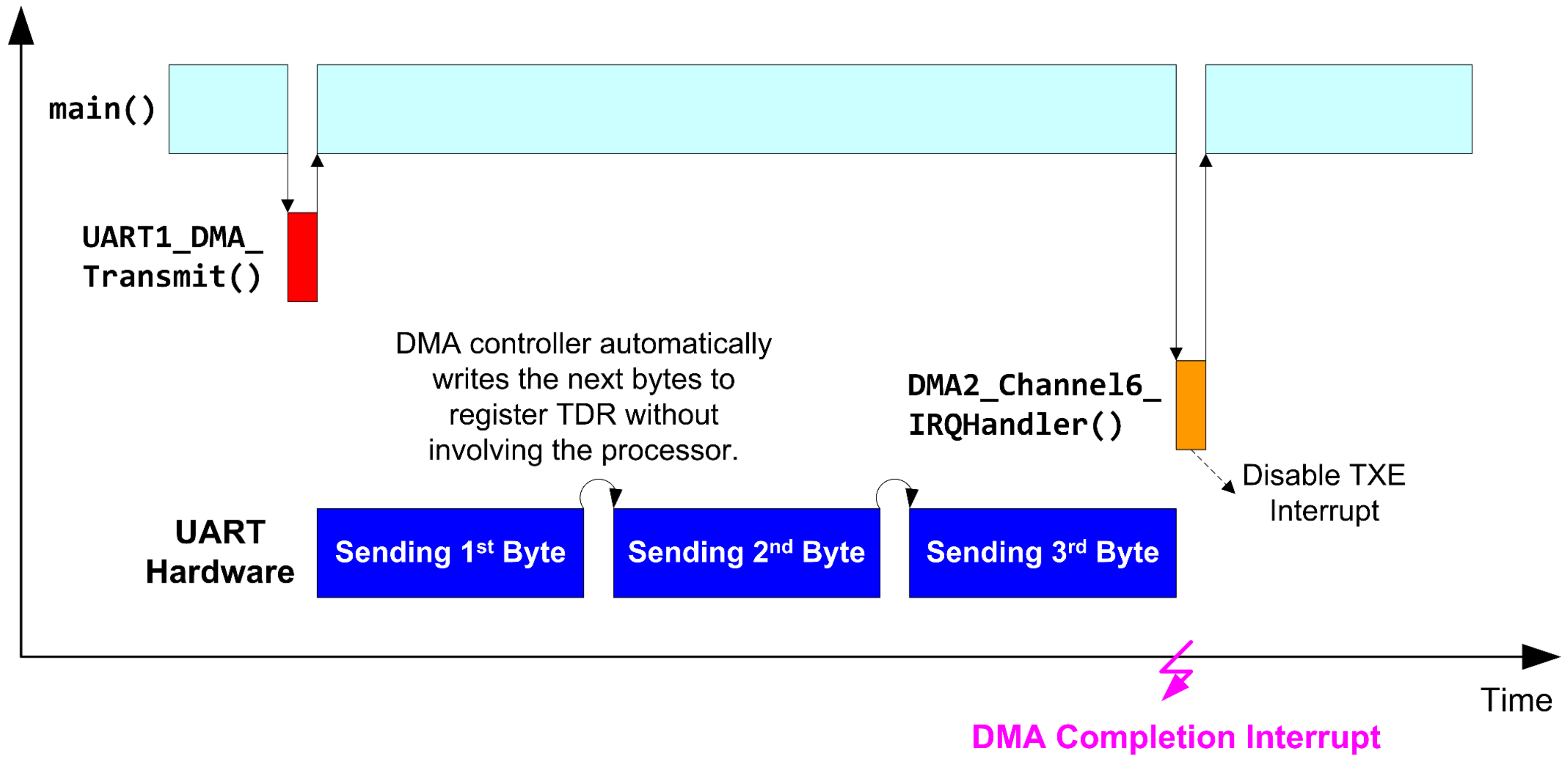
Transmit Buffer
TX_buffer[0]
TX_buffer[1]
TX_buffer[2]
TX_buffer[n-1]
DMA copies data from the address CMAR to the address CPAR
USART 1
TX pin
TX Shift Register
TDR Register
Hardware sets TXE flag if TDR is copied to TX shift register.
ISR Register
Hardware sets RXNE flag if RX shift register is copied to RDR.
RX pin
RX Shift Register
RDR Register
TXE triggers DMA request on channel 6
CPAR
CMAR
DMA 2 Channel 6
RXNE triggers DMA request on channel 7
DMA 2 Channel 7
CPAR
CMAR
NVIC Interrupt Controller
Channel 6 Interrupt Request
Channel 7 Interrupt Request
DMA copies data from the address CPAR to the address CMAR
Receive Buffer
RX_buffer[0]
RX_buffer[1]
RX_buffer[2]
RX_buffer[n-1]

```
UART_Transfer_Done_Flag = 0;

void DMA2_Channel6_IRQHandler(void) {
  ...
  // Check transfer complete flag (TCIF)
  if ( (DMA2->ISR & DMA_ISR_TCIF6)) {
    // Write 1 to clear the TCIF flag
    DMA2->IFCR |= DMA_IFCR_CTCIF6;
    UART_Transfer_Done_Flag = 1;
  }
  ...
}
```

```
void DMA2_Channel7_IRQHandler(void) {
  ...
  // Check transfer complete flag (TCIF)
  if ( (DMA2->ISR & DMA_ISR_TCIF7)) {
    // Write 1 to clear the TCIF flag
    DMA2->IFCR |= DMA_IFCR_CTCIF7;
    Lightweight_data_processing();
  }
  ...
}
```

# UART DMA: Receiving & Sending


main()
UART1_DMA_Transmit()
DMA controller automatically writes the next bytes to register TDR without involving the processor.
DMA2_Channel6_IRQHandler()
Disable TXE Interrupt
UART Hardware
Sending 1st Byte
Sending 2nd Byte
Sending 3rd Byte
Time
DMA Completion Interrupt